Google Cloud GDG

Kubernetes 101: an introductory workshop

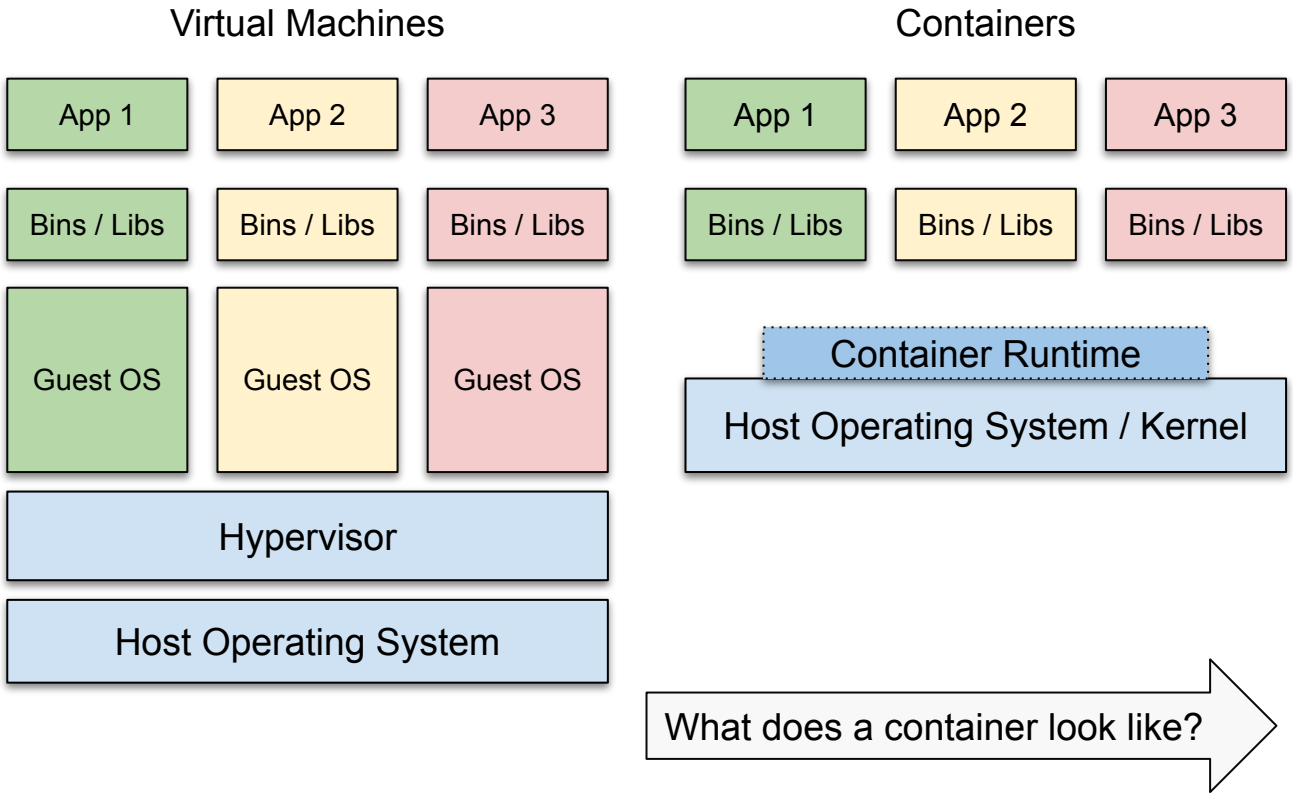
Carlos Afonso
Customer Engineer, Google Cloud

October 2023

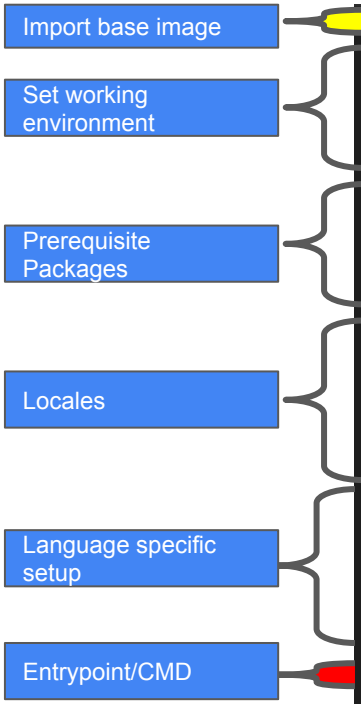
Why Containers ?

Containers vs. Virtual Machines (VMs)

- Consistent Environment
- Run Anywhere
- Isolation

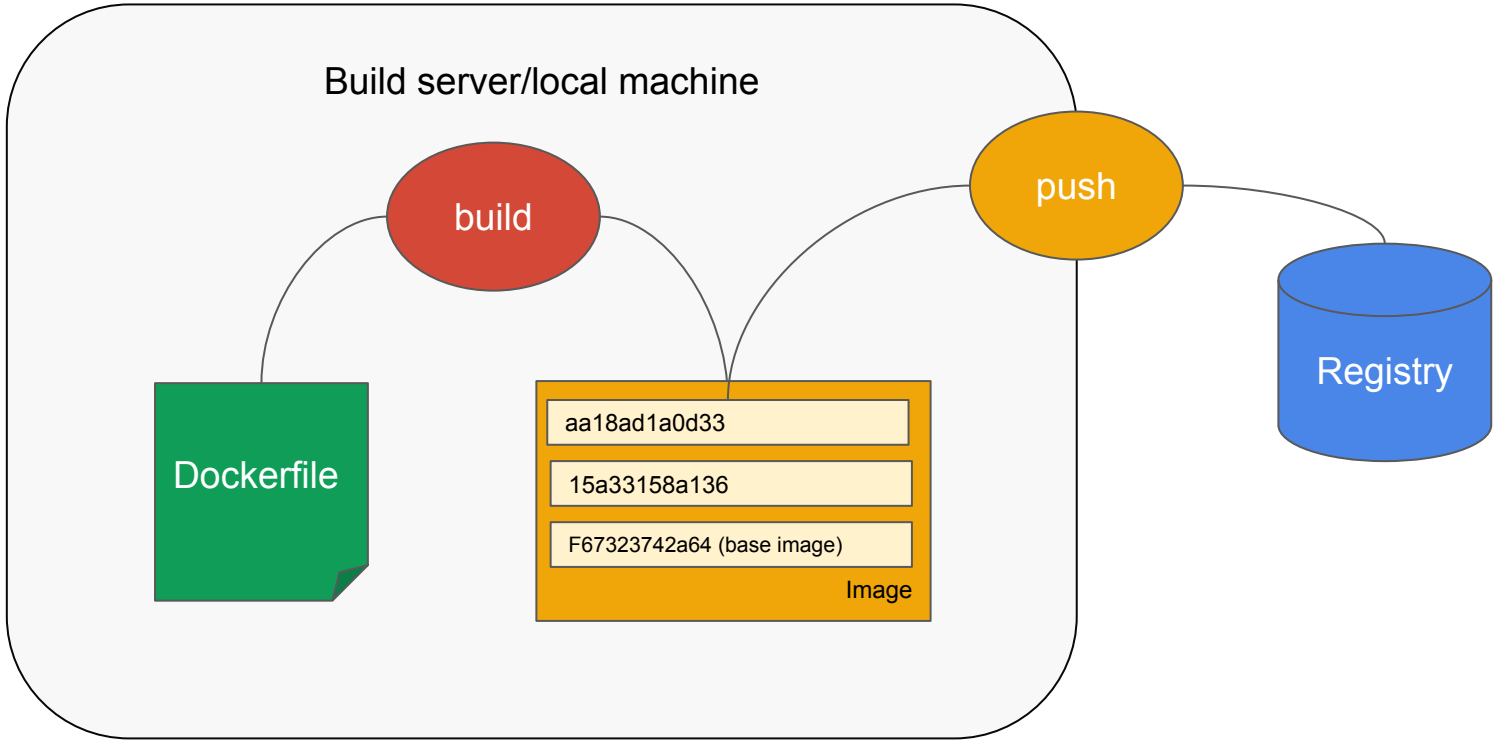


Anatomy of Dockerfile

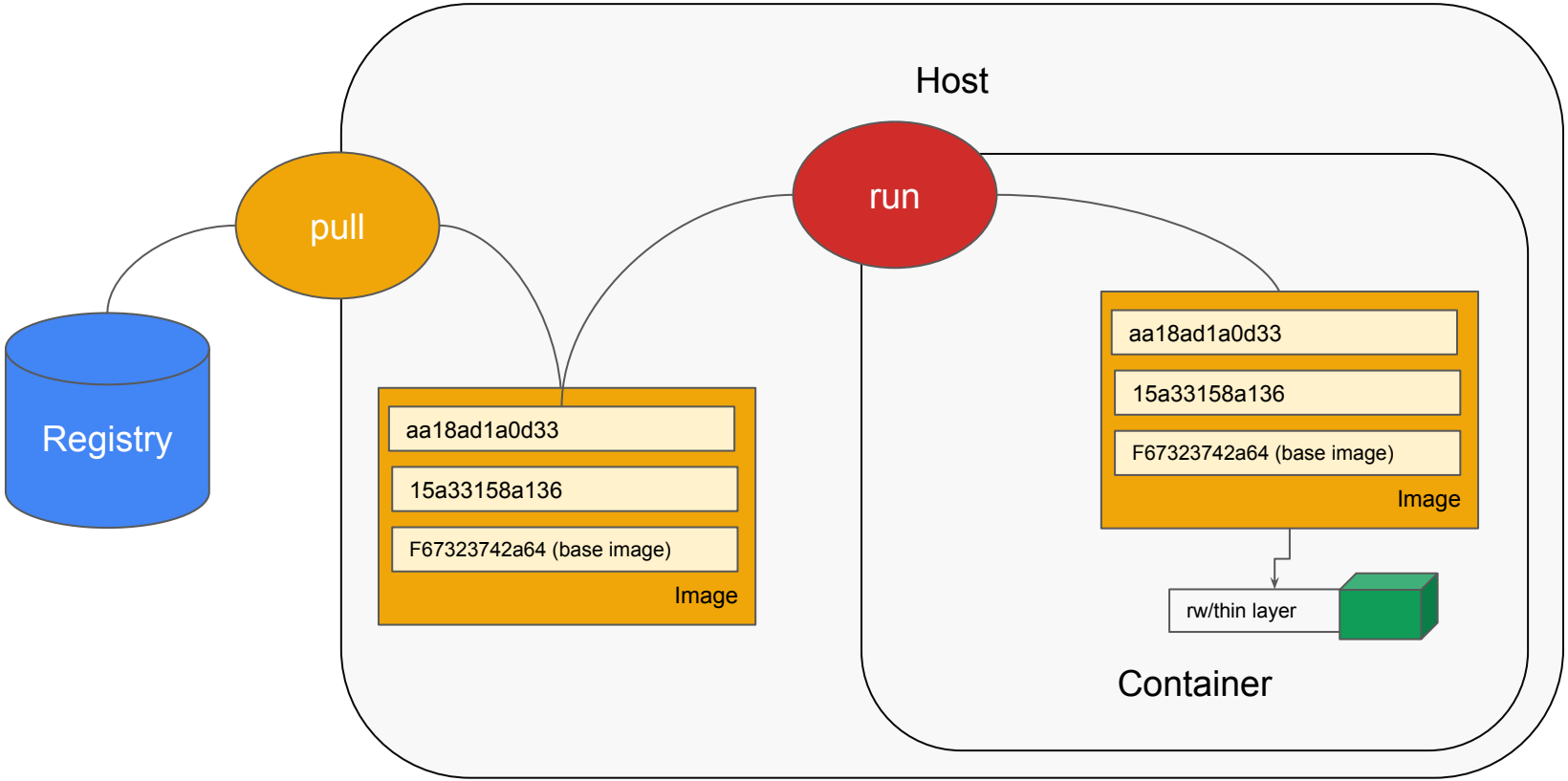



```
1  FROM debian:latest
2  # Work dir
3  RUN mkdir -p /usr/src/app
4  COPY . /usr/src/app
5  WORKDIR /usr/src/app
6  # packages
7  RUN apt-get update -qq
8  RUN apt-get install -qy  git rubygems ruby-dev linux-headers-* build-essential  zlib1g-dev locales
9  # Language locales
10 RUN locale-gen en_US.UTF-8
11 RUN localedef -i en_US -f UTF-8 en_US.UTF-8
12 ENV LANG en_US.UTF-8
13 ENV LANGUAGE en_US.UTF-8
14 ENV LC_ALL en_US.UTF-8
15 # Jekyll setup
16 RUN gem install jekyll bundler sass
17 RUN gem which bundler
18 RUN bundle --version
19 RUN bundle install
20 CMD ["bundle", "exec", "jekyll", "serve", "-H", "0.0.0.0", "--incremental", "--watch"]
```

Building an Image



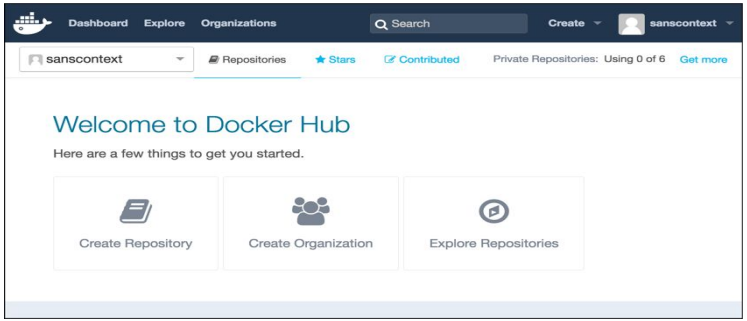
Running an Image

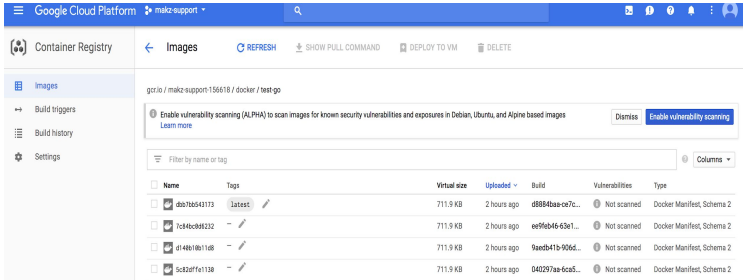


Container Registry

Docker Hub is a cloud-based registry service which allows linking to code repositories, build images, test them, and also stores manually pushed images.

Google Cloud has its own registry: **Artifact Registry**.





Why Kubernetes ?

Borg - Container management at Google

Large-scale cluster management at Google with Borg

Abhishek Verma† Luis Pedrosa‡ Madhukar Korupolu
David Oppenheimer Eric Tune John Wilkes
Google Inc.

Abstract

Google's Borg system is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different applications, across a number of clusters each with up to tens of thousands of machines.

It achieves high utilization by combining admission control, efficient task-packing, over-commitment, and machine sharing with process-level performance isolation. It supports high-availability applications with runtime features that minimize fault-recovery time, and scheduling policies that reduce the probability of correlated failures. Borg simplifies life for its users by offering a declarative job specification language, name service integration, real-time job monitoring, and tools to analyze and simulate system behavior.

We present a summary of the Borg system architecture and features, important design decisions, a quantitative analysis of some of its policy decisions, and a qualitative examination of lessons learned from a decade of operational experience with it.

1. Introduction

The cluster management system we internally call Borg admits, schedules, starts, restarts, and monitors the full range of applications that Google runs. This paper explains how.

Borg provides three main benefits: it (1) hides the details of resource management and failure handling so its users can focus on application development instead; (2) operates with very high reliability and availability, and supports applications that do the same; and (3) lets us run workloads across tens of thousands of machines effectively. Borg is not the first system to address these issues, but it's one of the few operating at this scale, with this degree of resiliency and com-

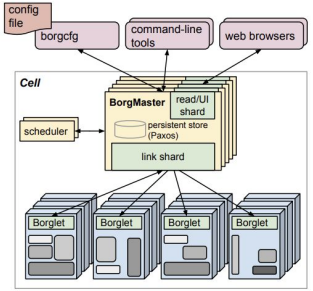


Figure 1: The high-level architecture of Borg. *Only a tiny fraction of the thousands of worker nodes are shown.*

cluding with a set of qualitative observations we have made from operating Borg in production for more than a decade.

2. The user perspective

Borg's users are Google developers and system administrators (site reliability engineers or SREs) that run Google's applications and services. Users submit their work to Borg in the form of *jobs*, each of which consists of one or more *tasks* that all run the same program (binary). Each job runs in one Borg *cell*, a set of machines that are managed as a unit. The remainder of this section describes the main features exposed in the user view of Borg.

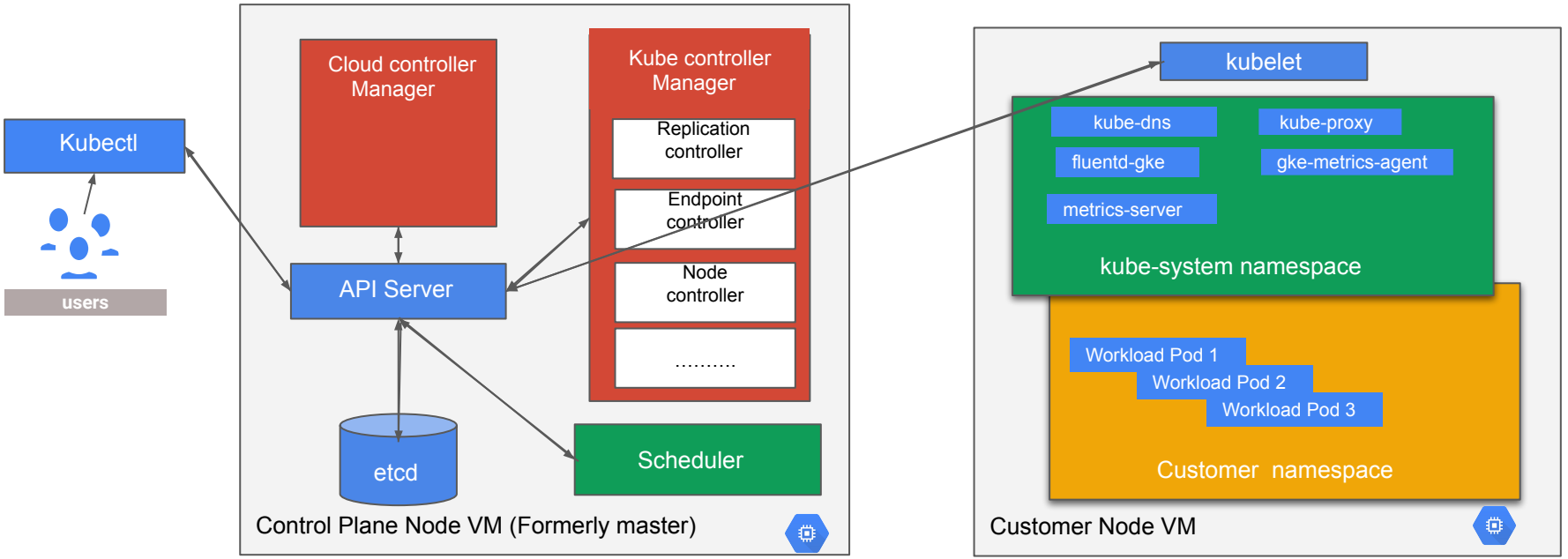
2.1 The workload

Kubernetes - Production Grade Container Orchestration

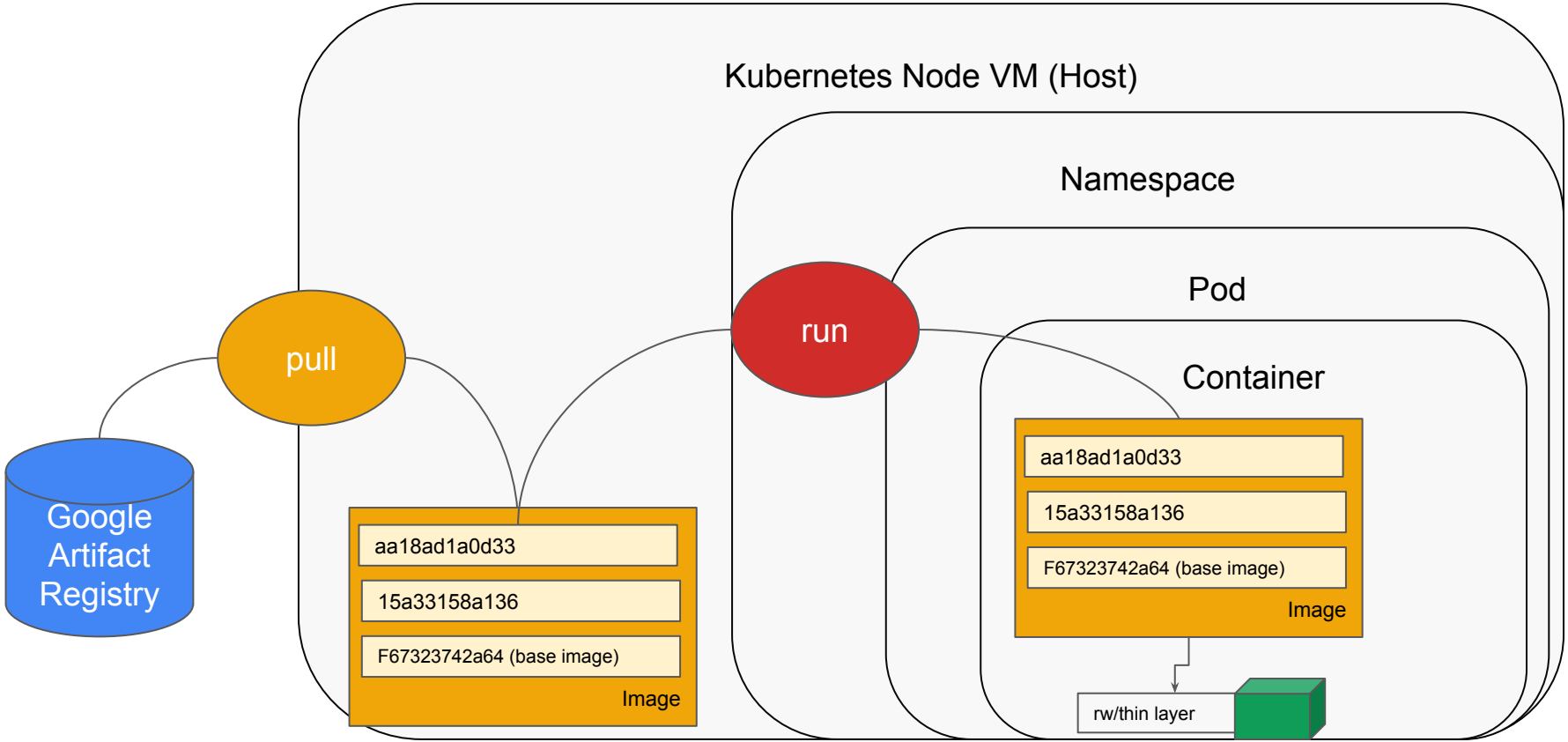
- Automated rollouts and roll backs
- Service health monitoring
- Automatic scaling of services
- Declarative management
- Deploy anywhere, including hybrid deployments

Cluster Architecture

Kubernetes - Cluster Architecture



Kubernetes - Node Architecture



API Object Definition

API object primitives include the following attributes:

- kind
- apiVersion
- metadata
- spec
- status

```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  s-k8sname: metric
spec:
  ports:
  - port: 2015
    protocol: TCP
    targetPort: 2015
  selector:
    run: metrics-k8s
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

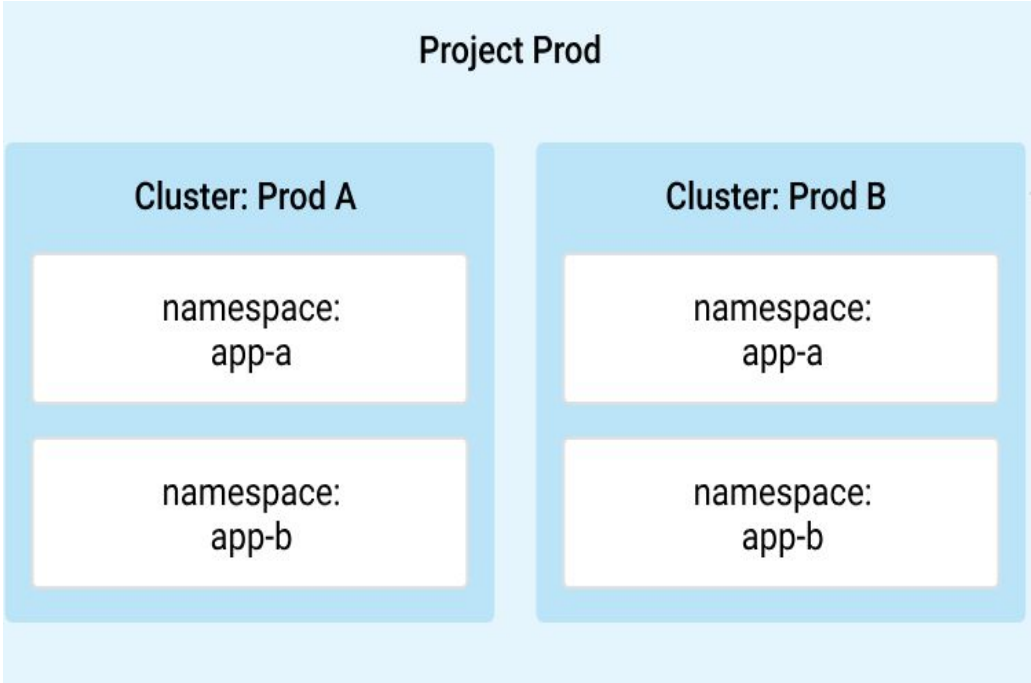
Workloads

Namespace

Namespace helps different projects, teams, or customers to share a cluster.

By default, cluster starts with the following three namespaces:

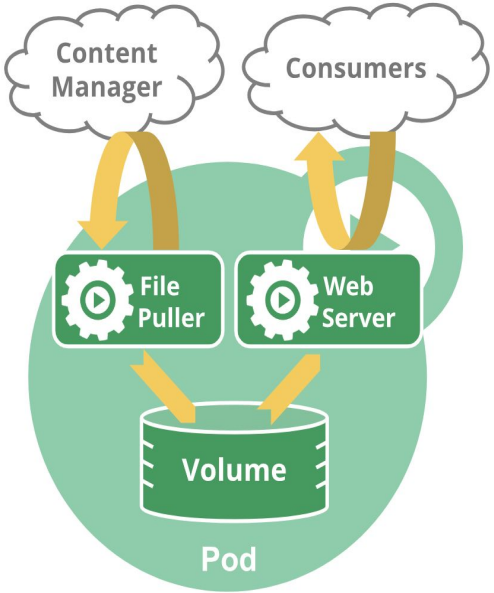
- Default
- Kube-public
- Kube-system



Pod

A Pod is the smallest, most basic deployable object in Kubernetes. It represents a single instance of a running process in the cluster. When a Pod runs multiple containers, the containers are managed as a single entity and share the Pod's resources such as:

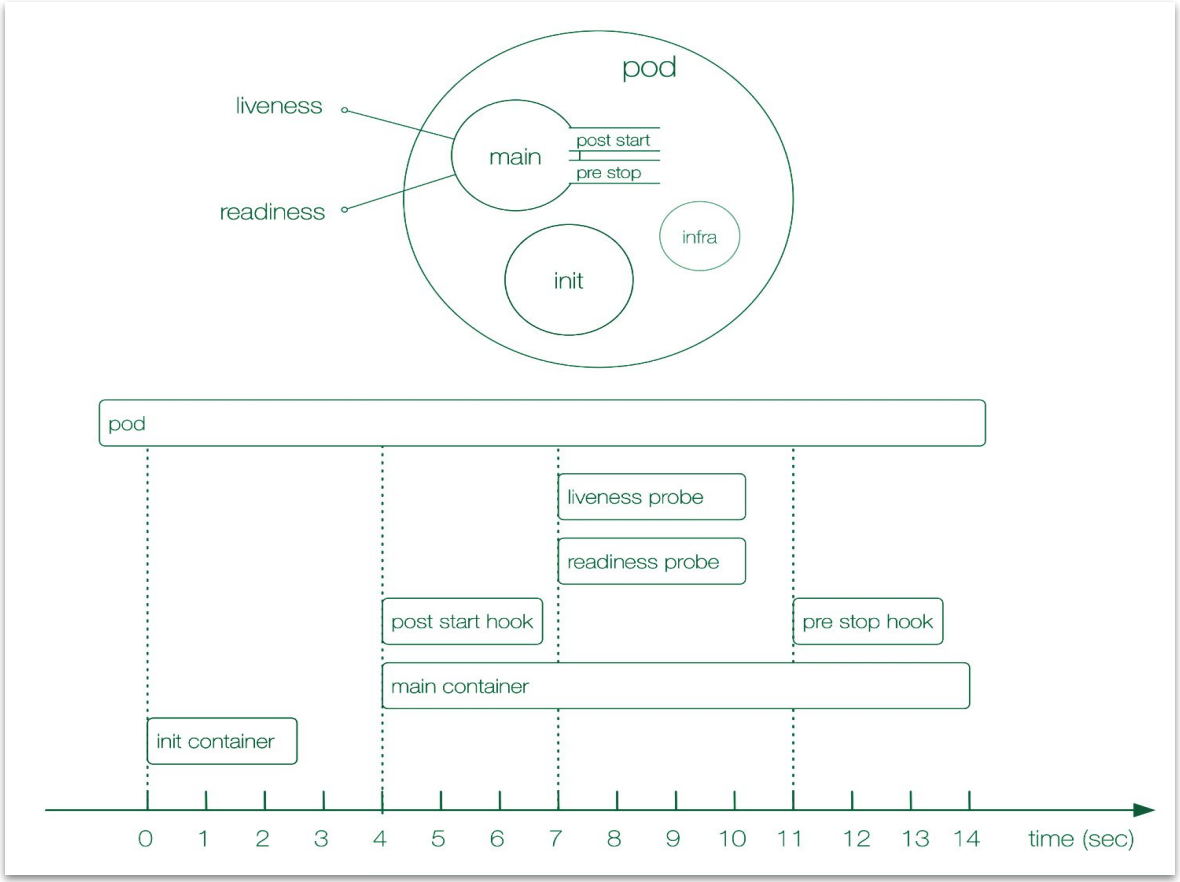
- Network
- Storage



pod diagram

A multi-container pod that contains a file puller and a web server that uses a persistent volume for shared storage between the containers.

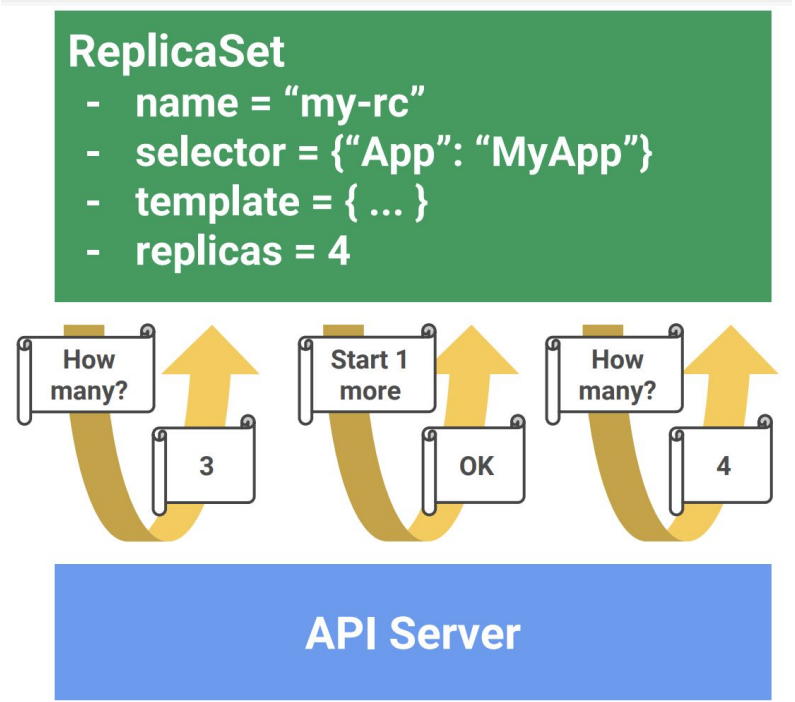
Life of a Pod


pod
liveness
readiness
main
post start
pre stop
infra
init
pod
liveness probe
readiness probe
post start hook
pre stop hook
main container
init container
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
time (sec)

ReplicaSets

A ReplicaSet is:

- Declaration of intent: run N copies of a pod
- Simple control loop
- One job: ensure N copies
  - Too few? -> start some
  - Too many? -> kill some
- Layered on top of Pods

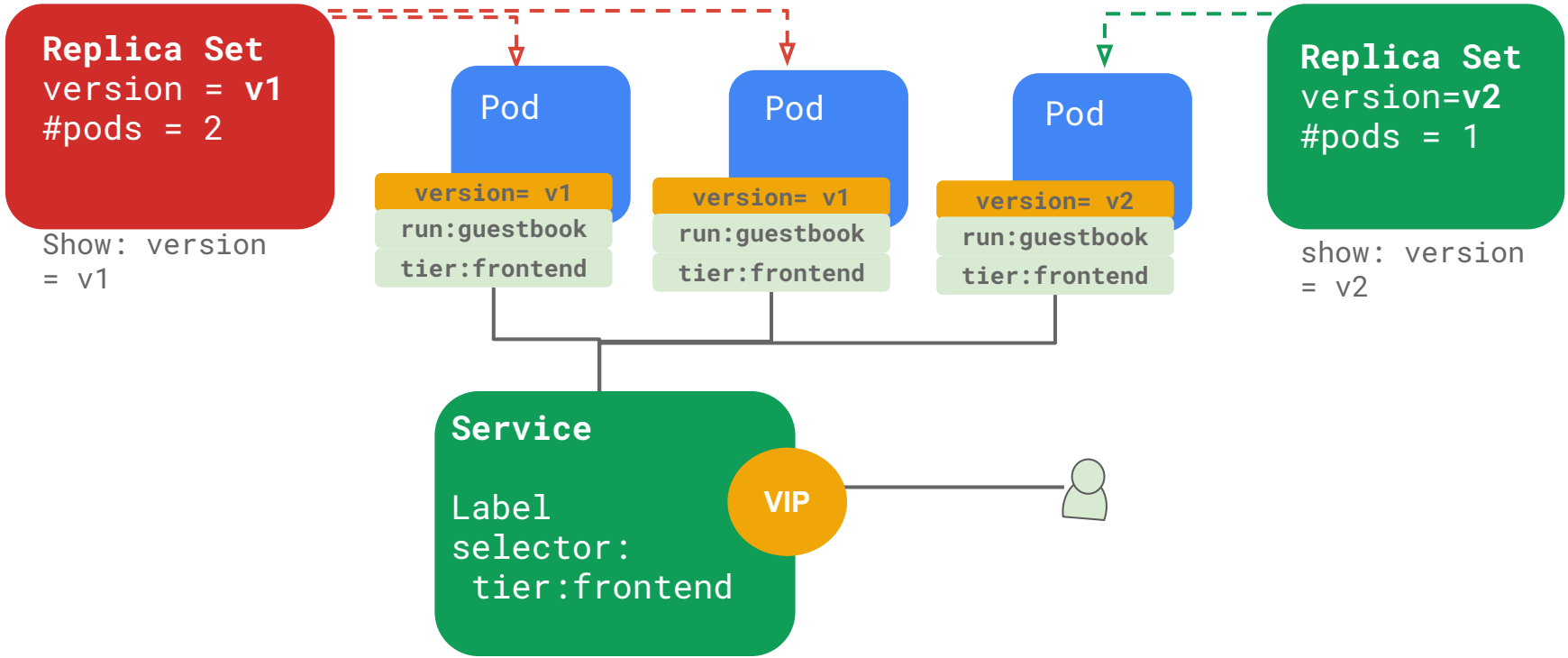


Deployment

A Deployment:

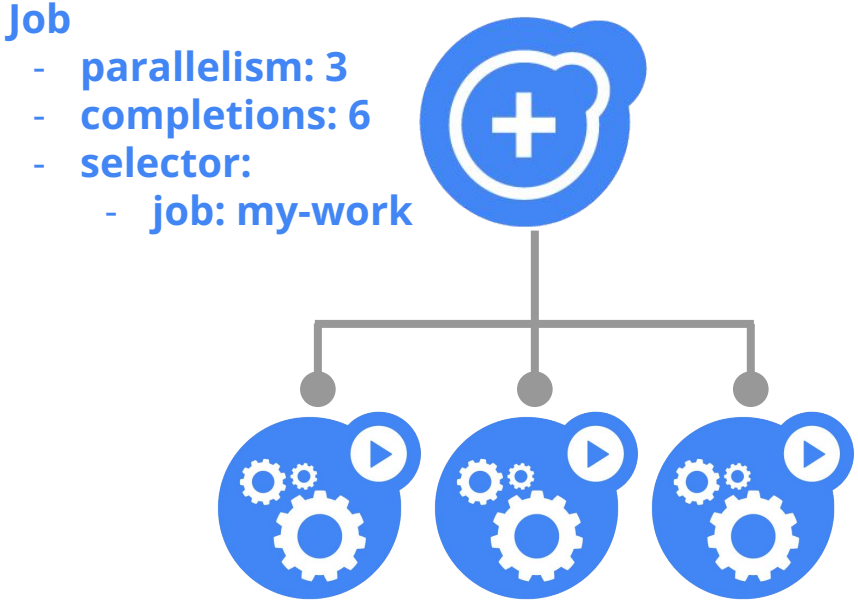
- Manages replica changes
  - Stable object name
  - Simply edit the object
  - Configurable server-side rolling-updates
- Can have multiple updates in flight
- Layered on top of ReplicaSets

Deployment - Rolling Update



Jobs

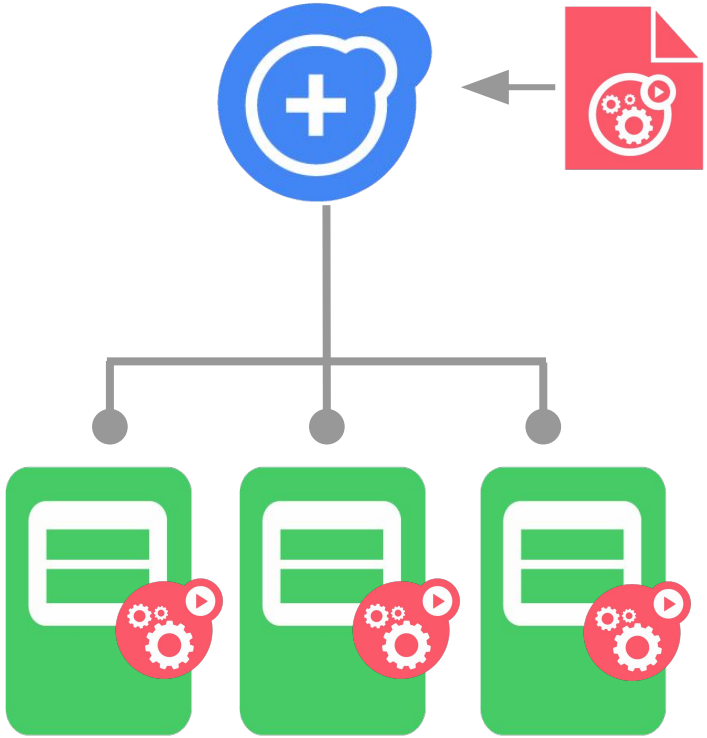
A Job creates one or more pods and ensures that a specified number of them successfully terminate. As pods successfully complete, the job tracks them. When a specified number of successful completions is reached, the job itself is completed. Deleting a Job cleans up the pods it created.



DaemonSets

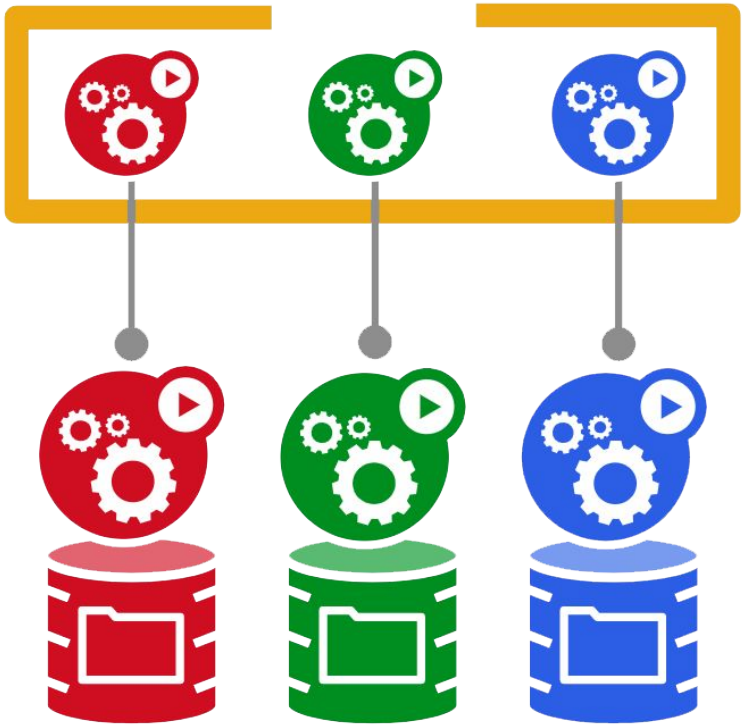
A DaemonSet runs a Pod on every node or a selected subset of nodes. It is not a fixed number of replicas which are created and deleted as nodes come and go. A DaemonSet is useful for running cluster-wide services such as:

- Logging agents
- Storage systems
- Run a node monitoring daemon on every node, such as Prometheus node exporter



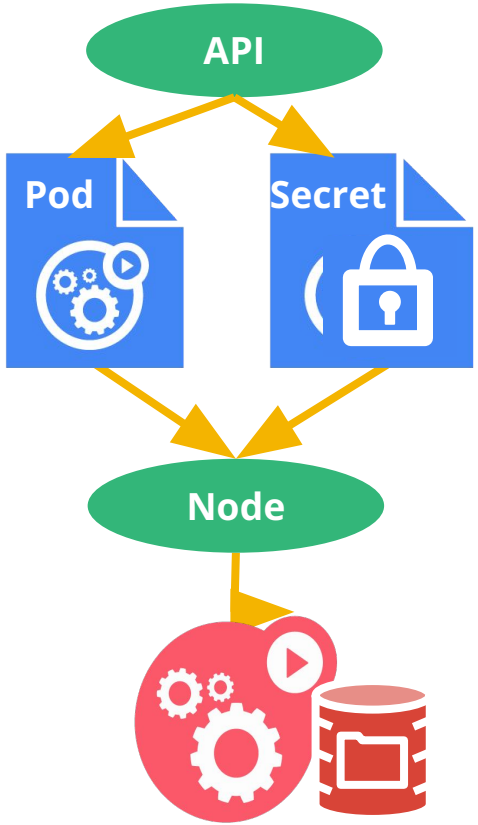
StatefulSets

A StatefulSet is a set of Pods with unique, persistent identities that have stable hostnames that Kubernetes Engine maintains regardless of where they are scheduled. State information and other resilient data is maintained in persistent disk storage. It is n ordinal index for the identity and ordering of their Pods. By default, StatefulSet Pods are deployed in sequential order and are terminated in reverse ordinal order.



Secrets

- How to grant a Pod access to a secured *something?*
  - **Secrets**: credentials, tokens, passwords, etc.
  - Do not put them in the container image
- Inject them as *virtual volumes* into Pods
  - Neither baked into images nor pod configs
  - Kept in memory so it never touches the disk
  - Not coupled to non-portable metadata API
  - Manage secrets via the Kubernetes API



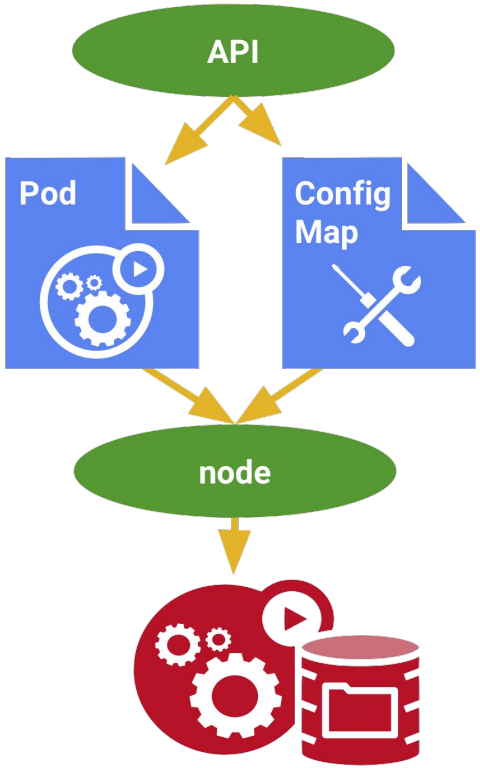
ConfigMaps

How to inject a configuration into a pod ?

ConfigMaps allow separating the configurations from the Pods and components, which:

- Keeps workloads portable
- Makes configurations easier to change and manage
- Prevents hardcoding configuration data to Pod specifications

They are useful for storing and sharing non-sensitive, unencrypted configuration information.



Services

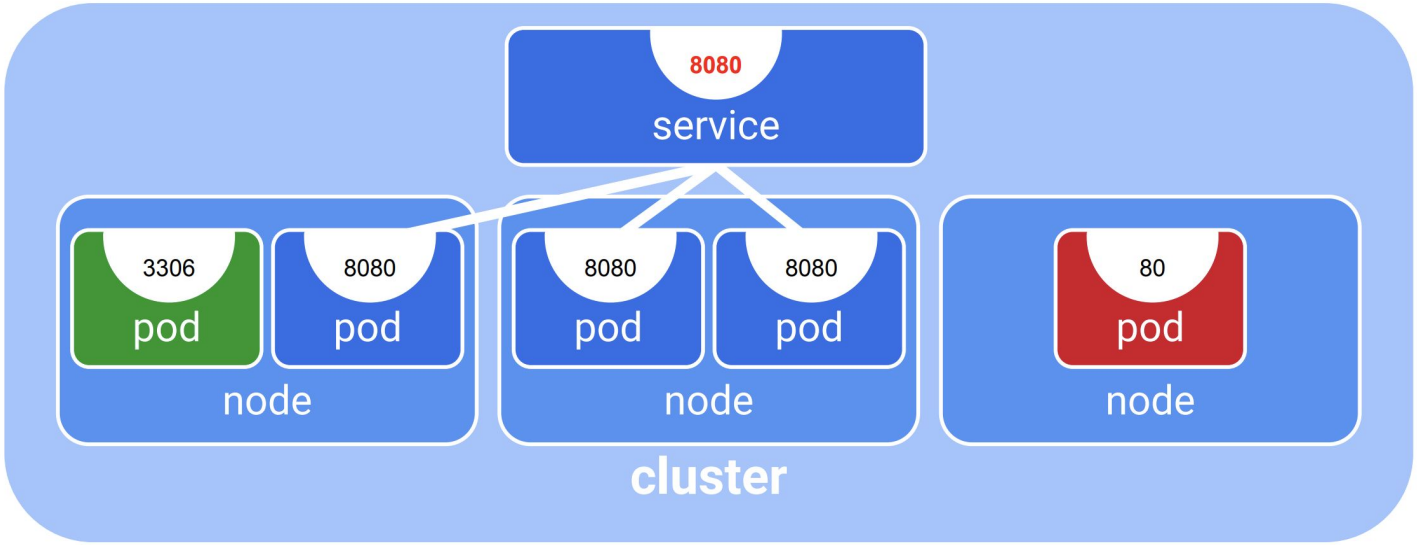
Services act as the unified method of accessing replicated pods. There are 4 major Service Types:

- ClusterIP (Default)
  - Exposes service on a strictly cluster-internal IP
- NodePort
  - Service exposed on each node's IP on a statically defined port
- LoadBalancer
  - Works in combination with cloud provider to expose service outside the cluster
- ExternalName
  - Reference endpoints outside the cluster by providing a static internally referenced DNS name

```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  name: metrics-k8s
spec:
  ports:
  - port: 2015
    protocol: TCP
    targetPort: 2015
  selector:
    run: metrics-k8s
  sessionAffinity: None
  type: Nodeport
status:
  loadBalancer: {}
```

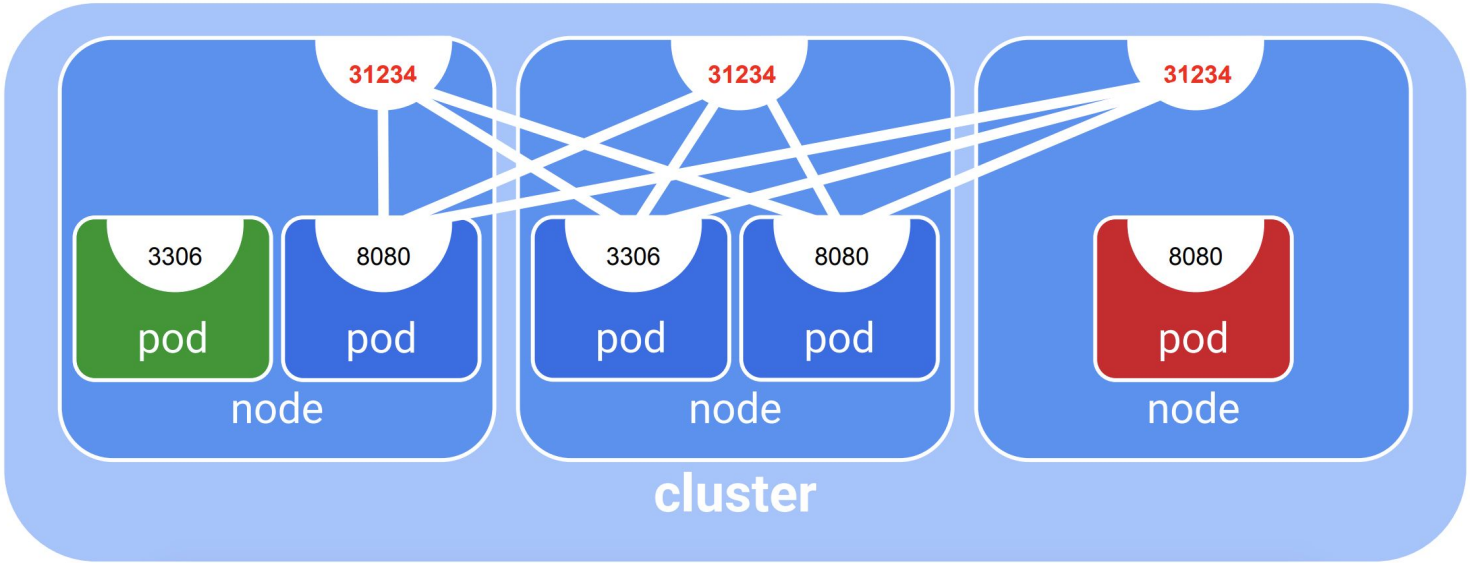
Services - ClusterIP

- Virtual port on the service VIP (internal to cluster)
- Service clients use this
- Stable ip:port
  - Does not change when pods move



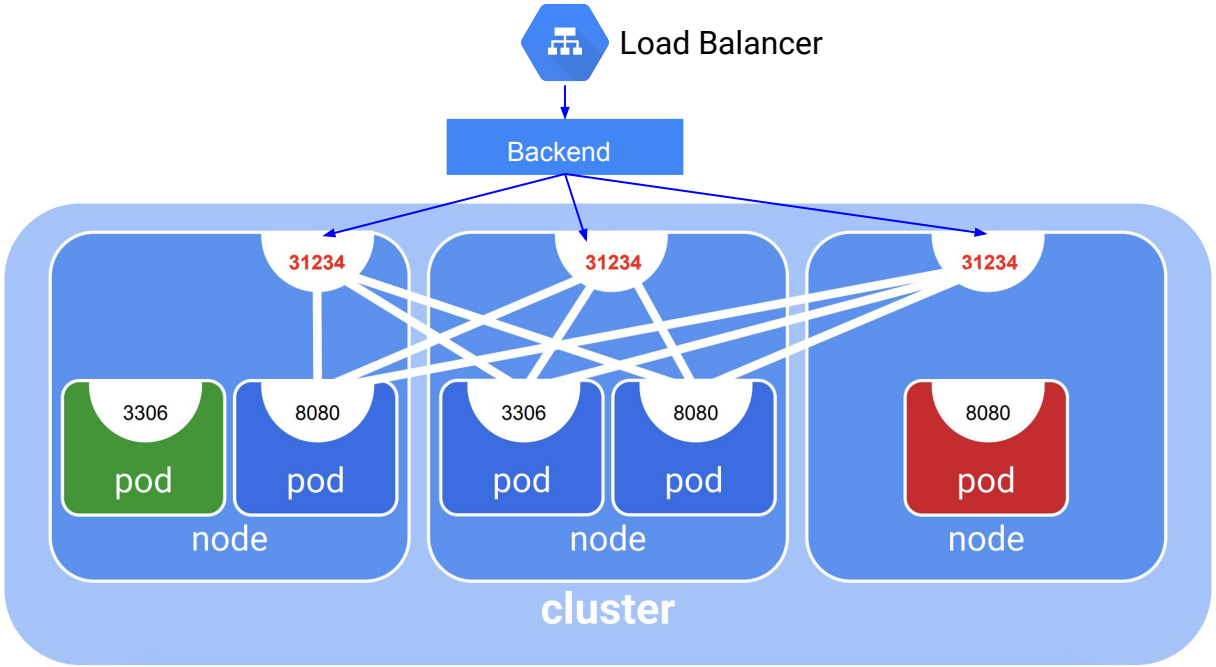
Services - Nodeport

- Maps a port on every node to a service port
- Allocates ports at random
- Interfaces with load-balancers that only understand nodes



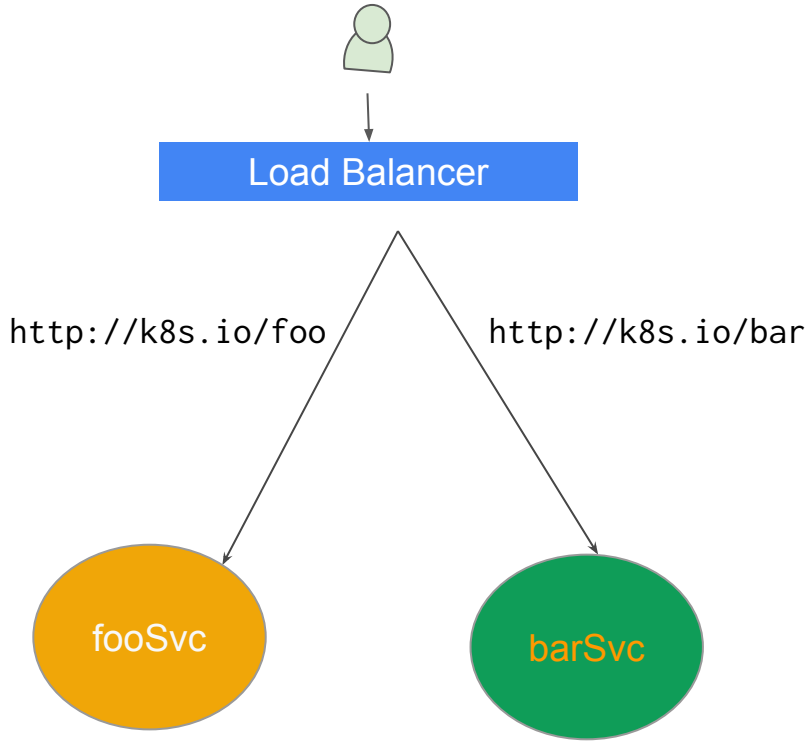
Services - Load Balancer

- Creates a TCP/UDP Network Load Balancing
- Regional external by default, internal by annotation
- Points traffic to the ClusterIP and NodePort



Services - Ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
spec:
  rules:
  - host: k8s.io
    http:
      paths:
      - path: /foo
        backend:
          serviceName: fooSvc
          servicePort: 80
      - path: /bar
        backend:
          serviceName: barSvc
          servicePort: 80
```

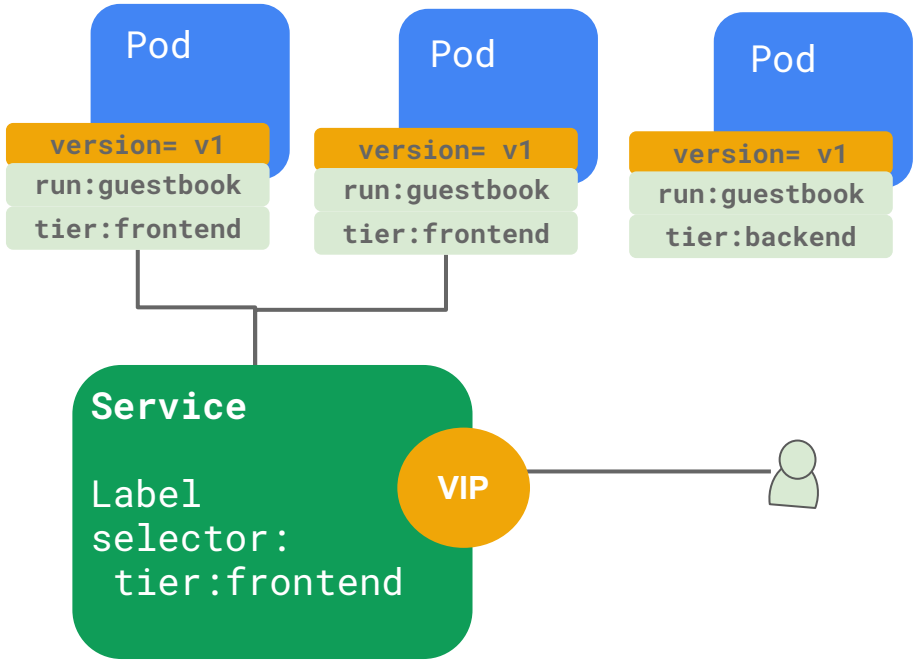


Services - Label Selectors

Labels are key/value pairs that are attached to objects, such as pods. They are intended to be used to specify identifying attributes of objects that are meaningful and relevant to users.

The client/ user can identify a set of objects via a label selector. There are 2 types of selectors:

- Equality-based
- Set-based

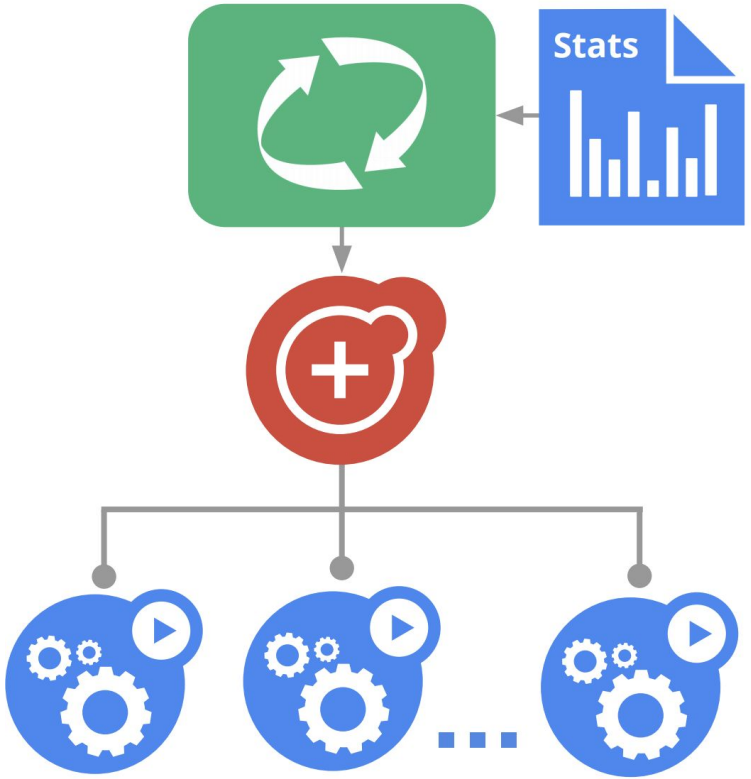


Break?

Next up: Kubernetes Concepts

Horizontal Pod Autoscaler

The Horizontal Pod Autoscaler automatically scales the number of pods as needed based on CPU / memory utilization, custom metrics like pubsub metric and external metrics pushed to stackdriver. It provides immediate efficiency and capacity when needed, operates within user-defined minimum/maximum bounds, and allows users to set it and forget it.



Assign Pod to a Node

- Scheduler will automatically do a reasonable placement to:
  - Spread pods across nodes
  - Not place the pod on a node with insufficient free resources, etc.
- When the user wants more control on a node where a pod lands, so as to:
  - Ensure a pod ends up on a machine with an SSD attached to it
  - Co-locate pods from two different services that communicate a lot into the same availability zone
- Achieved using NodeSelector

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  nodeSelector:
    disktype: ssd
```

Taints and Tolerations

Taints and tolerations work together to ensure that pods are not scheduled onto inappropriate nodes. One or more taints are applied to a node, which marks the node should not accept any pods that do not tolerate the taints.

- **`kubectl taint nodes node1 key=value:NoSchedule` -** places a taint on node **node1**

Tolerations are applied to pods and allow (but do not require) the pods to schedule onto nodes with matching taints, which specifies a toleration for a pod in the PodSpec.

```
tolerations:
- key: "key"
  operator: "Equal"
  value: "value"
  effect: "NoSchedule"
```

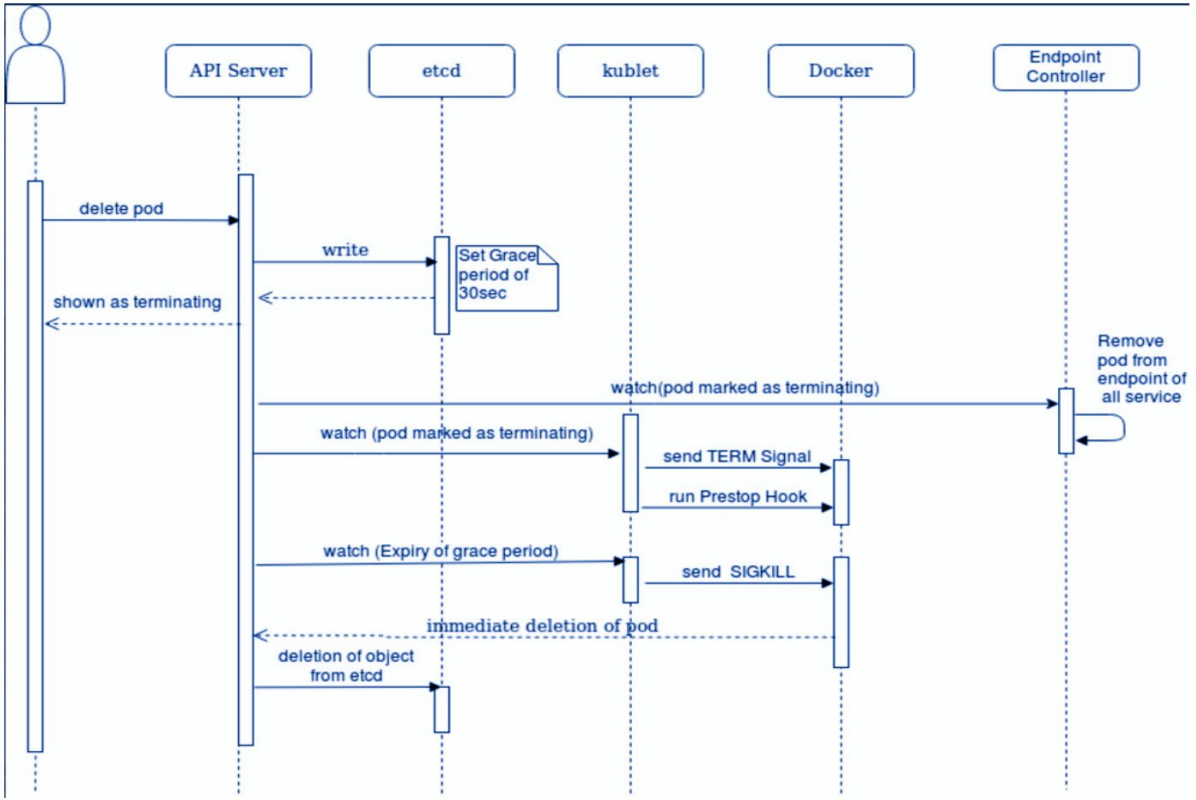
Pod Affinity and Anti-Affinity

Pod affinity and pod anti-affinity allow the user to specify rules about how pods should be placed relative to other pods. The rules are defined using custom labels on nodes and label selectors specified in pods.

```
apiVersion: v1
kind: Pod
metadata:
  name: with-pod-affinity
spec:
  affinity:
    podAffinity:

requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: security
            operator: In
            values:
            - S1
        topologyKey:
failure-domain.beta.kubernetes.io/zone
  containers:
  - name: with-pod-affinity
    image: gcr.io/hello-pod
```

Graceful Termination



Why GKE ?

Kubernetes - Cluster

- Kubernetes clusters as a service
- Runs Kubernetes on GCE
- Integrated with GCP
- Supports heterogeneous and multi-zone clusters
- Manages Kubernetes (auto-upgrades, scaling, healing, monitoring, backup, etc.)

GKE Concepts

Node Pools

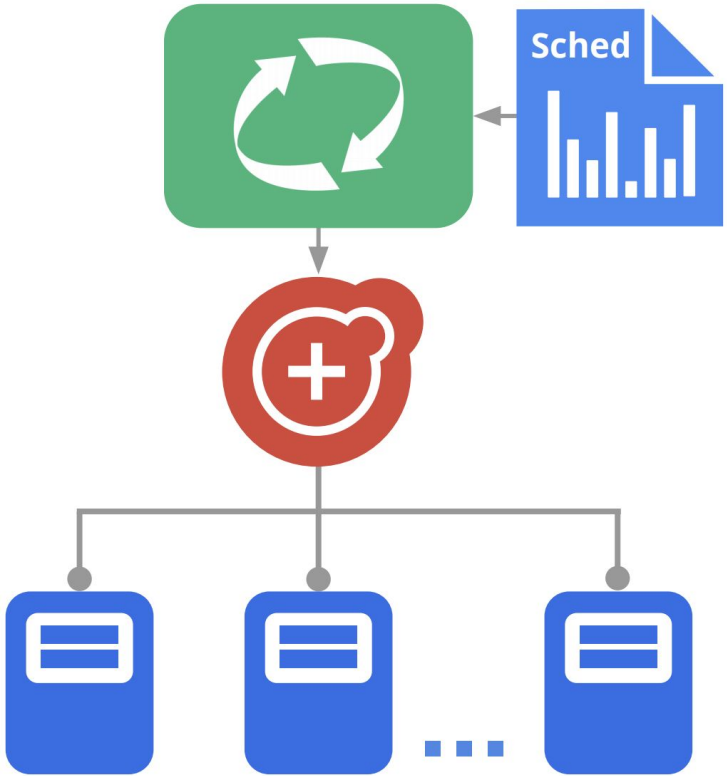
A **node pool** is a subset of node instances within a cluster that all have the same configuration and uses a NodeConfig specification.

Each node in the pool has a Kubernetes node label: **cloud.google.com/gke-nodepool** which has the node pool's name as its value.

The user can create, upgrade, and delete node pools individually without affecting the whole cluster using the gcloud container node-pools command.

Cluster Autoscaler

Cluster Autoscaler automatically scales the number of nodes as needed based on the scheduler backlog and idleness. It manages current efficiency, adds capacity as needed, operates within user-defined minimum/ maximum bounds, and allows users to set it and forget it.



Auto Upgrades

Node auto upgrades help keep the nodes in the cluster up to date with the latest stable version of Kubernetes. The upgrades use the same update mechanism as manual node upgrades. Some benefits of using auto upgrades are:

- Lower management overhead
- Better security
- Ease of use

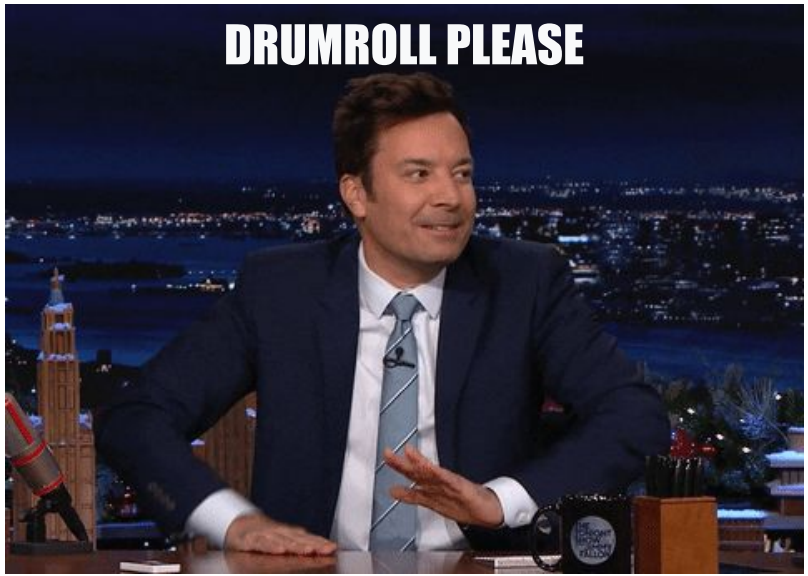
Please note: node auto upgrades are **not** available for Alpha Clusters or clusters running the Ubuntu node image.

Auto Repairing

The user can enable node auto repairing on a per-node pool basis. GKE uses the node's health status to determine if a node needs to be repaired. A node reporting a Ready status is considered healthy. GKE triggers a repair action if a node reports consecutive **unhealthy** status reports for a given time threshold, such as:

- NotReady status on consecutive checks over the given time threshold (approximately 10 minutes)
- Not reporting any status at all over the given time threshold (approximately 10 minutes)
- Node's boot disk is out of disk space for an extended time period (approximately 30 minutes)

Labs!



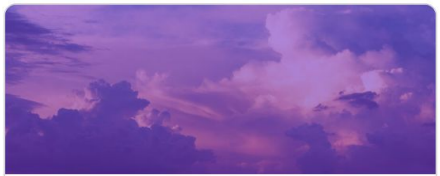
For our next trick, we'll need...

Google Chrome

Incognito Mode

ce.qwiklabs.com

Claim your GCP account in 4 easy steps



SCHEDULED COURSE

GDGs Spain - Kubernetes Workshop

Oct 21, 2023 12:00AM PDT

Girona

Labs

Deploy, Scale, and Update Your Website on Google Kubernetes Engine

Migrating a Monolithic Website to Microservices on Google Kubernetes Engine

Start Lab 02:00:00

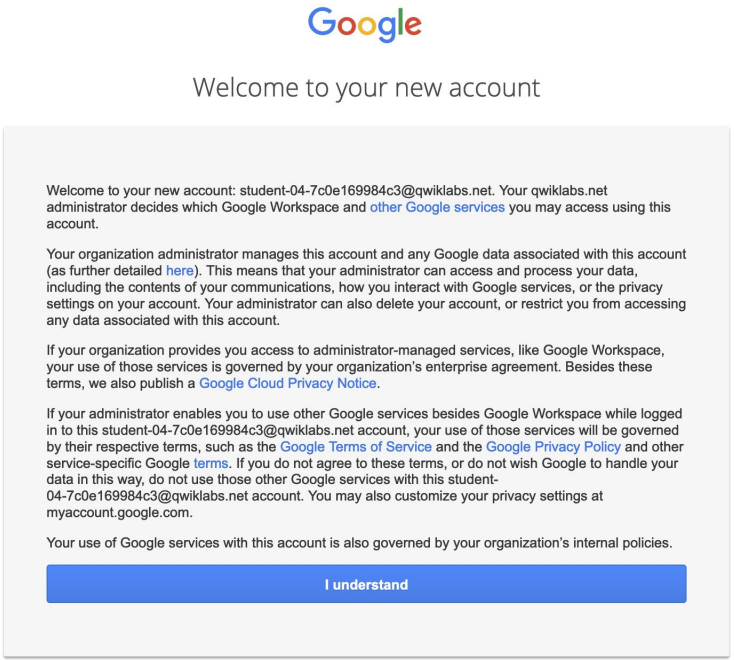
Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Open Google Cloud Console

User

student-04-7c0e169984c3

… actually, 5 easy steps



Questions?

Thank you.

Google Cloud